Data Analytics, Autumn 2024
Assignment 2
DEADLINE - 29.10.24 – 11:59 PM (Hard Deadline)

The task of this assignment is to implement the Association Rule Mining algorithm to analyze transaction data and discover interesting patterns between items. You will work with a dataset of transactions, and the goal is to generate **frequent itemsets** and extract **strong association rules** based on support, confidence, and lift measures.

The dataset captures the grocery purchases of multiple customers, with each row corresponding to a distinct transaction. In each row, the items bought during that transaction are listed as comma-separated values. Given this dataset, perform the following tasks. You may handle missing values (if any) according to a scheme of your choice.

I. Implement the **Apriori** algorithm from scratch to discover frequent itemsets.
   A. Allow the user to set a minimum **support** threshold as input. Output all frequent itemsets along with their corresponding support values.
   B. For the frequent itemsets found, generate association rules. Allow the user to specify a minimum **confidence** threshold. For each rule, calculate and display the following metrics: **a)** Support, **b)** Confidence, **c)** Lift. Display only the rules that meet the minimum confidence threshold. The input metrics are given by the user.
   C. Implement a feature to visualize: **a)** The top-N frequent itemsets using a bar chart. **b)** The top-N strongest rules (sorted by lift) using a scatter plot or any suitable visualization. Provide a detailed report summarizing the frequent patterns and the association rules discovered from the dataset.
   D. **Bonus**: Allow the user to specify additional interestingness measures such as Leverage or Conviction and implement the same.

II. Try to improve efficiency and/or performance of the algorithm. In the report explain the changes you've made and the reasons behind them. Marks will be given based on the **creativity of the solution**. Some ideas (**not compulsory** to implement all):
   1. You can consider custom support measures like Dynamic Minimum Support: Applies different support thresholds for different item categories (e.g., lower support for high-value items).
   2. You can implement custom interestingness measures. These could be based on (but not restricted to):
      A. Profitability: Weight rules based on the margin or profitability of the itemsets.

   B. Promotional Value: Rules that suggest potential cross-sell opportunities for underperforming or overstocked items.
   C. Actionability: Rules that are likely to lead to business decisions (e.g., improving product placement or marketing campaigns).

   Implement an efficient rule pruning strategy based on these measures to focus only on the most actionable and profitable rules.
3. You can use advanced data structures (e.g., Tree, Hash Tree) for faster itemset discovery.
4. You can explore different visualization of key itemsets and rules (e.g., heatmaps, itemsets graphs, customer-segmented rules).

Before evaluating the performance, we will first conduct a code **plagiarism** check. For evaluation, we will use different metrics to judge the efficiency of the algorithm. Additionally, we will measure the running time of each implementation.

For this assignment, each team must submit a **ZIP** file having a Python notebook (**.ipynb**) and a report **pdf** file. The notebook should include clearly labeled headings and proper comments for each cell for the execution. The report must have all the metrics, plots, and other important output values. Highlight any **advanced modifications** that you've added to the algorithm for better performance and why it would be beneficial for the task.

We'll use MS Teams to accept the submissions. Only **one member** from each team should submit the assignment deliverables. The report should have the **names** and **roll numbers** of all the team members.